UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 09/821,116 | 03/30/2001 | Michael N. Derr | P10559 | 3264 |

50890 7590 01/08/2008

CAVEN & AGHEVLI
c/o INTELLEVATE
P.O. BOX 52050
MINNEAPOLIS, MN 55402

| EXAMINER | |
|---|---|
| MAUNG, ZARNI | |
| ART UNIT | PAPER NUMBER |
| 2151 | |

| MAIL DATE | DELIVERY MODE |
|---|---|
| 01/08/2008 | PAPER |

**Please find below and/or attached an Office communication concerning this application or proceeding.**

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

**BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES**

Application Number: 09/821,116
Filing Date: March 30, 2001
Appellant(s): DERR, MICHAEL N.



Gregory D. Caldwell (Reg. No. 39,926)
For Appellant

**SUPPLEMENTAL EXAMINER'S ANSWER**

This is in response to the Order Returning Undocketed appeal to examiner mail on October 11, 2007, and the appeal brief filed on November 20, 2006 appealing from the Office action mailed March 29, 2206.

**(1) Real Party of Interest**

A statement identifying by name the real party in interest is contained in the brief.

**(2) Related Appeals and Interferences**

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

**(3) Status of Claims**

The statement of the status of claims contained in the brief is correct.

**(4) Status of Amendments After Final**

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

**(5) Summary of Claimed Subject Matter**

The summary of claimed subject matter contained in the brief is correct.

**(6) Grounds of Rejection to be Reviewed on Appeal**

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

**(7) Claims Appendix**

The copy of the appealed claims contained in the Appendix to the brief is correct.

**(8) Evidence Relied Upon**

| | | |
|---|---|---|
| 5,996,032 | BAKER | 11-1999 |
| 5,999,441 | RUNALDUE ET AL. | 12-1999 |

**(9) Grounds of Rejection**

The following ground(s) of rejection are applicable to the appealed claims:

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

Claims 2-3, 5-25 are rejected under 35 U.S.C. 103(a) as being unpatentable over Baker, U.S. Patent Number 5,996,032 (hereinafter Baker) in view of Runaldue et al., U.S. Patent Number 5,999,441 (hereinafter Runaldue).

Baker teaches a computer (12). Baker discloses the invention substantially as claimed. Taking claim 12 as an exemplary claim, Baker discloses a computer system (Fig. 1, computer 12) comprising: a processor subsystem supported by'bus 24 & 26 (Fig. 1); a device (18), which transfers data to or from, said processor subsystem (col. 10/lines 54-57, and col. 5/lines 35-39) and a device (14), which transfer data to/from said processor (col. 7/lines 38-48, col. 6/lines 3-14 and col. 21/lines 50-56); a controller

(20) adapted to control the transfer of data between said device and said subsystem (col. 5/lines 20-45); and writing in a register only the bits at the bit locations in the register for with the enable bit in the corresponding location in the bit enable field is set with the corresponding location in the data field (abstract, col. 10/lines 60-col. 11/line 20, col. 16/lines 44-52);

receiving a data value of a write command to a register in the controller (abstract); processing "interpreting" bits of the data value as enable bits in a bit enable field which designate predetermine bits within the register to which data is to be written and writing data only to the predetermined bits in a register write operation using a single write enable command (column 3, lines 15-27); however, Baker does not explicitly teach where the number of enable bits in the bit enable field is the same as the number of bits in the register where the data is to be written. Runaldue teaches writing individual bits of data to a "register" memory (10) (col. I/lines14-17, 60-64 and col. 2/lines 51-56), said method comprising: receiving bits of input data in a data field (DATA [0:7]) to be stored in said register, the register composed of memory cells (12) arranged in eight columns [0:7] (14), thereby the number of bits in said input data field being equal to the number of bits in the register and bit locations in the data field corresponding respectively to "bit locations" addresses in the register (Fig. 1, and col. 3/lines 30-57); receiving enable bits in a bit enable field (BIT_EN) from logic (18), the number of enable bits in the bit enable field being equal to the number of bits in the register and "bit locations" addresses in the bit enable field corresponding respectively to "bit locations" addresses in the register (col. 3/lines 65-col. 4/line 5, col. 5/lines 28-29

and col. 1/lines 19-41), and overwriting only the bits at the bit locations of the register according to the write enable bits in the write enable field (WRTDAT) for which the enable bit in the corresponding location in the bit enable field is set with the bit of input data in the corresponding location in the data field (col. 2/lines 7-12, 20-32). It would have been obvious to one ordinary skilled in the art at the time the invention was made given the suggestion of Runaldue for applying the his teachings to application using random access memory for storing data having multiple configuration lengths, the applicability to Baker environment including PC cards, i.e. memory cards, e.g., a SRAM (Static Random Access Memory) card, ROM (Read Only Memory) card, using IEEE 1394 standard communication (and suggesting the use of other different types of communication buses), typically used for communicating via telephone lines, a LAN card for connecting PCs via LAN, a SCSI (Small Computer System Interface) card for connecting to a SCSI apparatus, a sound card for playing music or producing sound effects by a PC, an ISDN (Integrated Services Digital Network) card for connecting to ISDN lines, and a Video Capture card for capturing a video signal. Motivation to combine would be modify data, such as individual bits of an addressed word within a single clock, reducing latency.

As per claim 13, an interconnecting device/means "bridge" (20 or 34) between the processor subsystem and at least said device, the controller being included in the switch or multiplexing function device (Baker: col. 30/lines 22-54 Fig. 26a).

As per claim 14, wherein the device comprises an storage device (Baker: 22) and he bridge comprises an "I/O controller hub" (ICH), which controls an IDE data, transfer between the processor subsystem and the IDE storage device (Baker: col. 30/lines 22-54 Fig. 26a).

Regarding claim 15, this claim is substantially the same as claim 10, same rationale of rejection 'is applicable.

Regarding claim 16, this claim comprises the software program stored in a tangible medium, said program, when executed, causing a computer to execute a method of claim 1, discussed above, same rationale of rejection is applicable.

Regarding claim 17, wherein said software program comprises a driver in the operating system software executed by a processor subsystem in the computer (Baker: host processor 44).

Regarding claims 18-19, this claim is substantially the same as combined limitations claims 1-3, and 9-10, same rationale of rejection is applicable

Regarding claims 2-3, wherein' the register is a "control" register for a data transfer operation (Baker: 58 on Fig. 2 control and status registers), including transfers data to or from and storage device (Baker: data transfer to or from said processor subsystem col. 10/lines 54-57, and col. 5/lines 35-39) and data transfer to/from said processor col. 7/lines 38-48, col. 6/lines 3-14 and col. 21/lines 50-56).

Regarding claim 5, wherein the control register is an "IDE DMA" status register (Baker: 76 on Fig. 2), and wherein the control register is a command register (Baker: col. 17/line 66-col. 18/line 50).

Regarding claims 6-8, wherein some of the bits of said register are not overwritten (Runaldue: col. 2/lines 25-32), wherein the data field and the bit enable field are received in parallel (Runaldue; col. 2/lines 14-25) and wherein the data field is provided at an address which is contiguous with the address for the bit enable field (Runaldue: 7-bit address signal (ADDR) [0:7] of Fig. 1, i.e. continuous).

Regarding claim 9, wherein the data transfer operation comprises an IDE data transfer between a processor subsystem and an external IDE storage device (137) (Runaldue: Fig. 2).

Regarding claim 10, wherein the processor subsystem posts an entire command sequence for setting up the IDE data transfer (Baker: col. 14/lines 64-66).

Regarding claim 11, wherein the method is carried out in controller in a interconnecting device or means connected between the processor subsystem and the external IDE storage device or peripheral supporting all data transfers therein (Baker: col. 6/lines 39-59).

Regarding claim 20, the combined teachings as discussed above, further teach receiving data of a single write command wherein the received data comprises a "data" input write data field (252) comprising an arbitrary number of bits, a write strobe signal

(264), and an address bit field [1:4] (266) of a register where data is to be written, and an write enable bit (258) comprising a predetermine number of bits, the outputted write command data (270) comprising: the address bit field (272) comprising a register address bit (274) and an individual bit select field (276) addresses bits (Fig. 10); modifying or changing the register with one or more bits of the data field that are associated with the enable bits of the bits enabled field (Baker: col. 16/line 32-col. 17/line 10).

Regarding claim 21, where the data (270) of the single write command comprises at least two bits (Baker: 1 to 4 GPIOs to be written, col. 16/lines 53-55), the write enable field (258) comprises one or more bit, and the input write data field comprises any number of bits (Baker: col. 16/lines 32-46).

Regarding claims 22-25, register has location in a logical space and a physical space (Runaldue: column 1, lines 19-41), single write command (Baker: abstract); write of data value to a space in the register (Baker: abstract); the data value comprises a number of enable bits that correspond to the same number of bits of the register (Runaldue: column 1, lines 14-41; column 2, lines 3-6, 49-56, and column 3, lines 50-57).

**Response to Arguments**

Regarding (section viii Arguments) **claim 20** (p. 6 of brief) it is argued that the applied references do not teach claim limitation, *receiving data of a write command*

*comprising a bit enable field and a data field, wherein the bit enable field and the data field comprises same number of bits in each field.*

Because according to appellant's interpretation of the applied reference(s), Baker appears to disclose in cited figure 10 and related description, a single write operation for writing an arbitrary number of data bits, including a register write circuitry which permits writing only to bits that must change in a register, while preserving the previous value of the remainder of bits, but *fails to describe where the bit enable field and the data field comprise the same number of bits.*

In response to the above-mentioned argument, Appellant's interpretation of the applied reference has been noted. The Baker reference has been reviewed. Baker discloses that the register write circuitry 250 shows *one* of an arbitrary number of similar circuits, each controlling a particular bit out of the arbitrary number or data bits. The write *enable bit field* (258) and the write *input data field* (252) are illustrated as input signals each of one bit value of their respective enable bit field and input data field signals as inputs to each register write circuitry.

Specifically, where the write *enable input signal (258)* is generated from the logic of an AND function (260), which receives as, inputs: a GPIO_ADR_OK signal (262), write_strobe standard signal (264) and GPIO_ADR [Ax] bit input signal (266). The AND function (260) provides the AND output (258) *to write enable bit (268) of data* register write flip-flop circuitry (250). When the *write enable input signal (258) is active*, the *input write data signal GPIO WRITE DATA [X]* (252) is written to the flip-flop (254) and appears on flip-flop output, namely, signal GPIO [X] (270). Only register write circuitry

flip-flops (250) with a 1 bit in their particular assigned address bit signal GPIO_ADR [Ax] will be written with their respective *GPIO WRITE DATA [X]* (252). AND function 260 provides the AND output 258 to *write enable bit 268 of data* flip-flop 250. Only flip-flops 250 with a *1 in their particular assigned address bit* [Ax] 266 *will be written with their respective GPIO write data [X]* 252.

Thus, the write *enable bit field* (258) and the write *input data field* (252) are illustrated as *input signals* each as *ONE bit value* inputted to *a register write circuitry (i.e. write command operation)*, this circuitry shows *one* of an *arbitrary number* of similar circuits, *each* controlling *a particular bit* out of the *arbitrary* number or data bits.

Baker teaches an *arbitrary number* of similar circuits, *each* controlling *a particular bit* outputted of the *arbitrary* number or data bits, each of the arbitrary number of circuits comprising a write *enable bit field* and a write *input data field* as input signals each as a bit value inputted to each register write circuitry of the arbitrary number of circuits.

Thus, an *arbitrary number* (N) of similar circuits, *each* controlling *a particular bit* outputted of the *arbitrary* number (N) of data bits, having N arbitrary number of write *enable bit fields* and N arbitrary number of write *input data field, the N* arbitrary circuits, *each* controlling *a particular bit* out of the *arbitrary* number N of data bits.

Baker teaches *where the bit enable field and the data field comprise the same number of bits* because the write circuitry receives as input one bit as an enable data field and one bit as an data field for each one bit of data to be written in the register, thus the write enable field and the write input data field have the same number of bits in

their fields for writes to a plurality of data register bits using a single register write operation.

Runaldue also teaches where *the bit enable field and the data field comprises same number of bits in each field.*

Specifically, teaching receiving bits of input data in a *data field (DATA [0:7])* to be written/stored in memory, and receiving enable bits in a *bit enable field (BIT_EN)* from logic (18), the number of enable bits in the bit enable field being equal to the number of bits in the register. Figure 3A illustrates one of alternative implementations of the bit enable decoder logic, which passes each bit of the 8-bit mask signal, MASK[0:7] to a corresponding one of the columns 14 to provide a bit-by-bit mask writing capability, illustrated in Figure 4A below, hence, the writing of each bit in an addressed word (W1) is controllable on a bit-by-bit basis based upon the corresponding bit in the mask word (column 5, lines 11-21).

Moreover, the control logic enables selected bits of an addressed word to be written to, without overwriting unselected bits in the addressed word. Hence, individual bits of an addressed word may be modified within a *single clock cycle*, *without* the necessity of performing a read-modify-write *sequence of operations* requiring at least three clock cycles plus external logic to manipulate the individual bits of the addressed word (column 2, lines 51-56).

Thus, Runaldue teaches a *single write operation* comprising an a *data field (DATA [0:7]) and* a *bit enable mask field (BIT_EN[0:7]), where* the bit enable field and the data field comprise the same number of bits.

Regarding (section viii Arguments) **claim 20** (p. 7 of brief) it is argued that the applied references do not teach *updating a register with one bit of the data field that are associated with enable bits of the bit enable field.*

In response to the above-mentioned argument, as mentioned above, Baker discloses a register write circuitry *writes* to a plurality of *data register bits* using a single register write operation using designated address bits and an address field for addressing a predetermined data register, where the designation address bits *designate predetermined bits within data register* to which *data is to be written* (see abstract). The write register circuitry discussed above, receives a write *enable bit field* signal and a write *input data field* signal each as *ONE bit value* inputted to *a register write circuitry (i.e. write command operation)*, this circuitry shows *one* of an *arbitrary number* of similar circuits, *each* controlling *a particular bit* out of the *arbitrary* number or data bits into a register.

Hence, Baker teaches writing (updating) in a register with one bit of the data field that are associated with enable bit of the bit enable field.

Runaldue discloses a memory having a prescribed number of bits defining a word length includes *memory cells* that *enable selective overwriting* on a bit-by-bit basis. Each memory cell includes a *logic gate* for generating a *gating signal* in response to *a supplied write signal and a bit enable signal.* The gating signal selectively connects the bistable latch of the memory cell to a voltage source to *enable storage of*

*the supplied data bit.* Hence, selected bits can be written to an address word, without overwriting the unselected bits in the stored word, *by supplying a mask signal that selectively drives the logic gates of the selected bits* (see abstract).

Hence, Runaldue teaches overwriting a register with one bit of the data field that are associated with enable bits of the bit enable field.

Regarding (section viii Arguments) **claim 20** (p. 7 of brief) it is argued that the applied references Runaldue that Runaldue appears to teach a bit enabled decoder logic 18 to enable bit by bit writing using data bits and mask bits, but there is *no indication that the mask bits are part of the data bits transmitted in a write command.* Further, Runaldue appears to be silent regarding *how exactly the mask bits are transmitted* and Runaldue thus does not teach *receiving data of a single write command* wherein the data comprises a bit enable field and a data field having same number of bits in each field as required by the Applicant's claim 20.

In response to the above-mentioned argument, Appellant's interpretation of the applied reference has been considered. However, the Boards attention is directed to the fact that, it is not clear where in claim 20 is the "transmission of a write command" recited nor where does claim 20 recite that "command data is transmitted".

Regarding arguments that it is unclear to Appellant that the *write command comprises the mask bit.* The teachings of Runaldue have been reviewed.

Runaldue discloses a memory having a prescribed number of bits defining a word length includes *memory cells* that *enable selective overwriting* on a bit-by-bit

basis. Each memory cell includes a *logic gate* for generating a gating signal in response to *a supplied write signal and a bit enable signal.* The *gating signal selectively* connects the bistable latch of the memory cell to a voltage source to *enable storage of the supplied data bit.* Hence, *selected bits can be written* to an address word, *without overwriting the unselected bits* in the stored word, *by supplying a mask signal* that selectively drives the logic gates of the selected bits (see abstract).

Hence, arguments that the *mask bits in Runaldue are not* "comprised in write command" have been fully considered but not found persuasive. In this case,

Runaldue further discloses where the logic circuit selectively causes to overwrite a stored value based upon the *write signal* and the corresponding *bit enable signal* (column 2, lines 25-32); the *logic gate* for generating a gating signal in response to a *supplied write signal and a bit enable signal*, the gating signal *causing* the corresponding memory cell to *overwrite a stored data value with a supplied data value*, and *control logic* for supplying the *bit enable signal* to a selected group of said columns in *response to a mask signal*; the *control logic enables* selected bits of an addressed word *to be written to*, without overwriting unselected bits in the addressed word. Hence, individual bits of an addressed word *may be modified within a single clock cycle*, *without* the necessity of performing a read-modify-*write sequence of operations* requiring at least three clock cycles plus external logic to manipulate the individual bits of the addressed word (column 2, lines 36-56);

Hence, [AS BEST UNDERSTOOD], the *logic circuitry* described by Runaldue implements a *write operation* and requires as input a *supplied write signal and a bit*

*enable signal*, for causing the gating signal to *overwrite a stored data value with a supplied data value; the logic circuitry further comprise a control logic* for supplying the *bit enable signal* to a selected group of said columns in *response to a mask signal*; the *control logic enables* selected bits of an addressed word *to be written to*, without overwriting unselected bits in the addressed word.

Arguments that it is unclear where in Runaldue does the "write command" comprise the mask bit", have been considered but not found persuasive.

Regarding (section viii Arguments) **claims 2, 3 and 5** (p. 8 of brief) it is argued that claims 2, 3 and 5 include claim 20 as a base claim. Appellant asserts that claims 2, 3 and 5 are allowable for at least the reasons stated above in regard to claim 20. Appellant submits that the above submissions are sufficient to overcome the present rejection of claims 2, 3 and 5 under Baker and Runaldue.

In response to the above-mentioned assertion, claims 2, 3 and 5 include claim 20 as a base claim. Accordingly, claims 2, 3 and 5 are unpatentable for at least the reasons stated above in regard to claim 20.

Regarding (section viii Arguments) **claim 6** (p. 8 of brief) it is argued that claim 6 requires "some of the bits of said register are not overwritten". According to Appellant, Runaldue does not teach where some of the bits of said register are not overwritten because "without overwriting unselected bits of the addressed word", of the Runaldue

reference is not the same as "some of the bits of said register are not overwritten" of Appellant's claim 6.

In response to the above-mentioned argument, Appellant's interpretation of the applied reference has been considered. In this case, according to the invention's disclosure: Hardware associated with the register receives a data packet containing the bit enable field and the data field "1010_1x0x" binary in the example of Figure 3 and overwrites the bit locations of the register for which the enable bit in the corresponding location of the bit enable field is set. *The other bit locations of the register are left unchanged.* In the example shown in Figure 3, bit locations 3 and 1 of the register are overwritten with the data in the corresponding bit locations of data field 302, while bit locations 2 and 0 of the register retain their initial values [see par 0034].

Thus, at least as described in the above specification, without limiting thereto, the invention overwrites the bit locations of the register for which the enable bit in the corresponding location of the bit enable field is set. *The other bit locations of the register are left unchanged.*

*Hence,* the "some bits of said register are not overwritten", seems to refer to the other bit locations of the register *which are not the bit locations* of the register *for which the enable bit in the corresponding location of the bit enable field.*

Runaldue discloses memory cells that enable selective overwriting on a bit-by-bit basis, where each memory cell includes a logic gate for generating a gating signal in response to a supplied write signal and a bit enable signal. The gating signal selectively enables storage of the supplied data bit. Hence, selected bits can be

written to an address word, *without overwriting the unselected bits in the stored word, by supplying a mask signal* that *selectively drives the* logic gates of the *selected bits* (abstract); enabling selected bits of a data word to be selectively written into a random access memory without overwriting *other non-selected bits of the data word stored in the memory* (column 2, lines 3-7).

Hence, *other* bit locations of the register which are *not the bit locations* of the register for which the enable bit in the corresponding location of the bit enable field is not found to be distinguishable over the prior art's unselected *bits of the data word stored in the memory*. Because the "some of the bits of said register not overwritten" seem to be *precisely and/or refer to, not the bit locations* of the register for which the enable bit in the corresponding location of the bit enable field.

The broadness reasonable interpretation of the claimed term/clause "some of the bits of said register not overwritten" does not exclude the non-selected *corresponding bit enable signal using a mask signal to write selected bits in the address memory word, without overwriting unselected bits of the addressed word.*

Thus, Runaldue teaches the enabling selected bits of a data word to be selectively written into memory without overwriting *other non-selected bits of the data word stored in the memory* (column 2, lines 3-7); *Selected bits can be written to an address word, without overwriting the unselected bits in the stored word*, by supplying a *mask signal* that *selectively* drives the logic gates of *the selected bits (abstract); the* logic circuit *selectively causes to overwrite a stored value based upon the write signal and the corresponding bit enable signal and* can be used to access a selected memory

location, while *using a mask signal to write selected bits in the address memory word, without overwriting unselected bits of the addressed word.*

Additionally, Baker teaches the designation address bit designates predetermined bits within the data register to which data is to be written as active and *other* bits within the data register as inactive. The method and system further include *writing data only to the predetermined bits in a register write operation* using a single write enable command.

Hence, the applied reference(s) teach claimed limitation, namely, "some of the bits of said register not overwritten".

Regarding (section viii Arguments) **claims 7-8** (p. 9-10 of brief) it is argued that claims 7-8 require that *the data field and the bit enable field are received simultaneously and at respective address contiguous to each other.* Because according to Appellant's interpretation, Runaldue neither discloses *anything of the bus level protocol (that is bit enable enclosed in the data phase) nor a system in which control register are written and* Appellant is unable to locate where Runaldue teaches, the data field and the bit enable field received simultaneously and at respective address contiguous to each other. Because "bistable latches" are not "data field" and "bit enable field"

In response to the above-mentioned argument, Appellant's interpretation of the applied reference has been reviewed. (i) It remains unclear where is a "*bus level protocol (that is bit enable enclosed in the data phase)" recited in claims 7-8.*

(ii) Regarding "writing to a control register". Baker teaches writing to a data register (see abstract) as discussed above in detail. Figure 2 in the Baker reference illustrates where data register (76) comprises a DMA control and status registers, where slave logic (66) when enabled r*esponds to write commands* at PCI memory *address ranges specified by base address registers contained in 68* and further posted write operations when enabled (column 7, lines 23-32). Thus, the applied reference teaches a "*system in which control register are written".*

(iii) Arguments that the applied reference(s) fail to teach wherein the data field and the bit enable field are received simultaneously (claim 7), and wherein the data field is provided at an address which is contiguous with the address for the bit enable field (claim 8) have bee fully considered.

In this case, Appellant failed to provide for dependent claims 7-8 argued separately, reference to the specification by page and line number, and to the drawing, if any, by reference characters. (see MPEP 1205 (a)1(v) of 37 CFR 41.37). Broadest reasonable interpretation will be applied (MPEP 2106/2111). However, although the specification

(iv) Claim 7, recites, wherein the data field and the bit enable field are received simultaneously; (v) claim 8, recites, wherein the data field is provided at an address which is contiguous with the address for the bit enable field.

According, to the specification, *hardware* associated with the register r*eceives a data packet* containing the bit enable field and the data field and overwrites the bit

locations of the register for which the enable bit in the corresponding location of the bit enable field is set. The other bit locations of the register are left unchanged [see par 0034]. [AS BEST UNDERSTOOD] "contiguous" and "simultaneous" refers to a data packet containing the bit enable and data field to perform the write operation on the register.

Runaldue teaches a memory cell including the circuitry (Fig. 2) for performing a write operation, which stores a data input, supplied on the input signal path (42). Specifically, where *simultaneous assertion of the write enable and bit enable signals* causes the bistable latch 40 to store the data supplied on the data input line 42 (column 4, lines 45-54).

Hence, [AS BEST UNDERSTOOD], the *logic circuitry* described by Runaldue implements a *write operation* and requires as input a *supplied write signal and a bit enable signal to be simultaneously asserted*, for causing the gating signal to *overwrite a stored data value with a supplied data value.*

Baker teaches writing data only to the predetermined bits in a register write operation using a single write enable command, as discussed above. Baker further teaches transmitting a write command packet by placing it on a bus. Specifically, the logic 64 implements the control required for the interface 20 to operate on PCI bus 24, which permits the operation of memory writes instructions. For the memory write operation, the interface 20 DMA (control register 76) write operation results in a PCI memory write, memory write line command on the PCI bus (column 7, lines 12-22). The slave logic 66 implements the control required for the interface 20 to operate on

the bus, which when enabled, logic 66 responds to memory write commands at PCI memory address ranges specified by base address registers contained in control registers 68. The logic 66 performs *posted write operations* when enabled (column 7, lines 23-32). Once the interface assumes control of the bus, the interface 20 can issues write commands on PCI bus 24 by specifying the appropriate address range in the controlling packet control list (column 20, lines 41-45). DMA engine (74) acquires a certain address for obtaining instructions when active, this permits the interface 20 to generate PCI memory commands (column 21, lines 11-15). The DMA controller/engine 74 is controlled by data structures called packet control lists or PCLs, which contains command information, which the DMA fetches from memory as needed. These commands tell the engine the sources and *destinations for the data* and how many bytes it is to transfer, some commands move chunks of data between the buffers and the bus (column 23, lines 52-column 24, line 3). The packet control list may be organized as a contiguous set of memory locations that contain the commands, control parameters, and data buffer pointers required by a DMA channel to transfer one IEEE 1394 data packet, or to move data between the bus (column 24, lines 4-16). Transfer command packet control list 0 (456 on Figure 20), includes a *load command 472, source address at location 474 and store command and location 476.* Load command 472 goes to data register 478 which supplies DMA register 480, where the store 1 command 476 provides input to memory location 482.

Baker discloses, as discussed above, the address of a register to which the write enable bit 258 (of Fig. 10) may be written to based on the address field. The address

field 272 of Fig. 11 includes the register address bits (274) and individual bit select field (276) addresses bits. The address field 272 permit addressing selectively specific bits individually or in different combinations (column 16, line 53 to column 17, line 10).

Thus, Baker teaches a single write command to be placed on the bus the write command comprising the *destinations for the data* and how many bytes it is to transfer, the command specifically comprises a packet control list that contain the commands, control parameters, and data buffer pointers required to transfer/move data packets. Transfer command packet control list (i.e. contiguous meaning next or near in time or sequence) includes a *load/store command and respective locations.* Baker teaches where the destination in the command of the register where data is to be written includes the register address bits and individual bit select field addresses bits that permit addressing selectively specific bits to be written in the register.

In this manner the combined teachings "contiguous" and "simultaneous" bit enable and data fields, which has been interpreted as referring to a data packet containing the bit enable and data field to perform the write operation on the register.

Regarding (section viii Arguments) **claims 9-11** (p. 11-12 of brief) it is argued that claims 9-11 require the processor subsystem posts the entire command sequence for setting up the data transfer.

In response to the above-mentioned argument, the applied references have been reviewed. In this case, Baker discloses that PCI-interface ASIC (20 of Figure 1) performs a primary function of controlling transfer of data packets between devices

operating in an environment that supports PCI bus (24) and devices operating in a high-speed input/output peripheral environment (column 5, line 15-34); Figure 2 which shows the functional partitioning of PCI-interface ASIC 20, thus interface 20 further comprising a PCI slave logic (66) which performs the control logic necessary for PCI-interface ASIC 20 to operate on the PCI bus as a slave device, which when enabled, PCI slave function 66 responds to memory write commands at PCI memory address ranges specified by base address registers contained in control and status registers (68) and performs slave burst transfers when enabled by the slave burst bit in the miscellaneous control register and further performs posted write operations when enabled by a control bit in the miscellaneous control register (column 7, lines 23-32); Once enabled as master on the PCI bus 24, PCI-interface ASIC 20 can issue write commands on PCI bus 24 by specifying the appropriate address range in the controlling packet control list (column 20, lines 41-49, where the structure of the transfer command packet is shown on Figure 20);

Thus, Baker teaches a subsystem (20) which posts (so called "entire command sequence") commands for performing (so called "setting up") the data transfer, e.g. memory write operation.

Appellant further argues (p. 11 of brief) that "Runaldue neither discloses *anything of the bus level protocol (that is bit enable enclosed in the data phase) nor a system in which control register are written".* This argument is the same presented

with respect to claims 7-8, discussed above, same response provided above to this argument is applicable.

Regarding (section viii Arguments) **claim 12** (p. 12-14 of brief) it is argued that claim 12 requires receiving a data value of a write directed to a control register, interpreting bits of the data value as a data field and as enable bits in bit enable field and the number of bits being equal to the number of bits in the control register. Appellant argues that although Baker teaches where part of the address filed is used to specify the bits to be loaded, Appellant is unable to located with the reference teaches "receiving a data value of a write directed to a control register".

In response to the above-mentioned argument, Appellant's interpretation of the applied references is noted. However, Appellant's argument that address bits (274) is not data, does not make sense, does this mean "bits" are not data ? Given that rationale, address bits are not storable nor transmittable since there not data, presuming Appellant concedes data can be transmitted or stored.

Back to the argued limitation, *receiving a data value of a write directed to a control register.* The apparatus (computer) claim 12 comprising a method further comprising "receiving a data value of a write directed to a control register".

The claim clause "data value of a write" has been interpreted as meaning a write having a data value, subsequence claim clause language shows the intended purpose or use of being directed to a register.

[AS BEST UNDERSTOOD], the *logic circuitry* described by Runaldue implements a *write operation* and requires as input a *supplied write signal and a bit enable signal*, for causing the gating signal to *overwrite a stored data value with a supplied data value; the logic circuitry further comprise a control logic* for supplying the *bit enable signal* to a selected group of said columns in *response to a mask signal*; the *control logic enables* selected bits of an addressed word *to be written to*, without overwriting unselected bits in the addressed word.

Thus, Runaldue teaches the logic circuitry of a write operation, which receives as input a supplied write signal (i.e. data value).

Baker teaches a register write circuitry for writing an arbitrary number of data register bits with a single register write operation, where when the write enable is active, input write data 252 is written to flip-flop 254 and appears on flip-flop output where only flip-flops with a 1 in their particular assigned address bit will be written with their respective GPIO write data 252 supplied to the flip-flop (Figure 10).

Thus, Baker teaches receiving a data value of a write (e.g. write data 252) directed to a register.

The Baker patent claims directing the single write command to a single predetermined bit (see claim 5), wherein a designation address bit designates predetermined bits within the data register to which data is to be written as active and other bits within the data register as inactive (Column 3, lines 17-27).

Baker further teaches a write command comprising data value. Specifically, as discussed with respect to claims 9-11, Baker teaches where the interface (20 of Figure

1) performs a primary function of controlling transfer of data packets between devices (column 5, line 15-34); the interface 20 as shown on Figure 2 comprising a logic (66) which performs the control logic necessary for interface 20 to operate on the PCI bus as a slave device, which when enabled, logic (66) responds to memory write commands at memory address ranges specified by base address registers contained in control registers (68) and performs data burst transfers when enabled by the slave burst bit in the miscellaneous control register and also further performs posted write operations when enabled by a control bit in the miscellaneous control register (column 7, lines 23-32); interface 20 controlled by logic 64 permits the operation of *memory writes, and memory write line instructions*, where for the memory write operation, the interface 20 DMA *write operation results* in a PCI memory write, memory write line command on the PCI bus (column 7, lines 12-22). PCI generates PCI memory command, and input/output commands using the DMA engine (74) (column 21, lines 11-15); interface 20 can issue write commands on bus by specifying the appropriate address range in the controlling packet control list (column 20, lines 41-49, where the structure of the transfer command packet is shown on Figure 20); the pack control lists or PCLs are data structures, which contains *command information which tell the DMA the sources and destinations for the data and how many bytes it is to transfer*, some commands *move chunks of data* between the IEEE 1394 transmit FIFOs and PCI bus 24, or between the general receive FIFO 80 and PCI bus 24. Another command *moves data* between PCI bus 24 and auxiliary port local bus 26. Other commands are

for secondary functions and are called auxiliary commands (column 23, lines 52-column 24, line 15)

Thus, Baker teaches a receiving a data value of a write command directed to a register.

Appellant further argues that the applied reference does not teach "interpreting bits of data". This argument has been fully considered. The portions cited by Appellant with respect to claim 12 have been reviewed. Namely, claim 12 relates to a computer to receive a data value of a write directed to a control register and interpreting bits of the data value as a data field 402 and bits of the data value as an enable field 401 (paragraphs [0034], lines 1-7 in page 14, and [0035], lines 8-10, in page 14, see Figure 4 (p. 3 of brief). Cited portion of page 14, lines 1-10 recites,

*Hardware associated with the register receives a data packet containing the bit enable field and the data field ("1010_l x 0x" binary in the example of Fig. 3) and overwrites the bit locations of the register for which the enable bit in the corresponding location of the bit enable field is set. The other bit locations of the register are left unchanged. In the example shown in Fig. 3, bit locations 3 and 1 of the register are overwritten with the data in the corresponding bit locations of data field 302, while bit locations 2 and 0 of the register retain their initial values.*

*Generally speaking (without reference to the example shown in Figure 3, the enable field allows any combination of N register bits to be over-written with a 2*N- bit*

*write command with relatively simple hardware implementation. The invention can be easily applied to existing hardware structures having existing control registers by providing an alternate register location for implementing the "bit-granular writes" as described above. In the case of an I/O Controller Hub (ICH) or other suitable hardware device, the alternate register location may be placed in memory space, thereby allowing the processor to post the bit-granular writes for use with the streamlined techniques described below.*

*Fig. 4 is a table illustrating an example in which individual bits in a control register are overwritten. The specification where further reviewed, however, the specification. do not seems to mention Figure 4.*

These portions provide NO indication as to how are bits "interpreted".

The applied references as mentioned above, teach hardware (register write circuitry) associated with the register including receives a the bit enable signal and the data signal and overwrites the bit locations of the register for which the enable bit in the corresponding location of the bit enable field is set, without overwriting the other bit locations of the register are left.

Appellant further argues (p. 13 of brief) that "Runaldue neither discloses *anything of the bus level protocol (that is bit enable enclosed in the data phase) nor a system in which control register are written".* This argument is the same presented

with respect to claims 7-8, discussed above, same response provided above to this argument is applicable.

Regarding (section viii Arguments) **claim 13-14** (p. 15 of brief) it is argued that claims 13-14 include claim 12 as a base claim. Hence, according to Appellant, claims 13-14 are allowable for at least the reasons stated above in regard to claim 12. Appellant believes the above is sufficient to overcome the present rejection of claims 13-14 under Baker and Runaldue. Applicant respectfully requests that the rejection of claims 13-14 be reversed.

In response to the above-mentioned argument, claims 13-14 are unpatentable for at least the reasons stated above in regard to claim 12.

Regarding (section viii Arguments) **claim 15** (p. 15 of brief) it is argued that claims 15 includes claim 12 as a base claim. Hence, according to Appellant claim15 are allowable for at least the reasons stated above in regard to claim 12. Moreover, claim 15 requires the processor subsystem posts an entire command sequence in the controller for setting up the IDE (integrated drive electronics) data transfer. Because, according to Appellant, Baker in column 14, lines 64-66, discloses that a check is made to determine whether the command is receive, transmit, PCI to/from local bus or auxiliary command. Also, Runaldue does not appear to teach that the processor subsystem posts an entire command sequence in the controller for setting up the IDE (integrated drive electronics) data transfer,

as required by the Applicant's claim 15. However, Appellant is unable to locate where Runaldue or Baker teaches the processor subsystem posts an entire command sequence in the controller for setting up the IDE (integrated drive electronics) data transfer.

In response to the above-mentioned argument, this claim limitation, applying the broadest reasonable interpretation is substantially the same argued limitation with respect to claims 9-11, namely, a subsystem (called processor subsystem) that posts a sequence, command or instruction (called entire command sequence) in the controller for performing (called "setting up" the transfer (called IDE "integrated drive electronics" data transfer), same rationale of rejection is applicable.

As discussed above, Baker further teaches where the interface (20 of Figure 1) performs a primary function of controlling transfer of data packets between devices (column 5, line 15-34); the interface 20 operates on the bus, logic 66 responds to memory write commands at memory address ranges specified by base address registers contained in control registers (68) and also further performs posted write operations (column 7, lines 23-32); interface 20 controlled by logic 64 permits the operation of *memory writes, and memory write line instructions*, where for the memory write operation, the interface 20 DMA *write operation results* in a PCI memory write, memory write line command on the PCI bus (column 7, lines 12-22). PCI generates PCI memory command, and input/output commands using the DMA engine (74) (column 21, lines 11-15); interface 20 can issue write commands on bus by specifying the appropriate address range in the controlling packet control list (column 20, lines 41-49,

where the structure of the transfer command packet is shown on Figure 20); the pack control lists or PCLs are data structures, which contains *command information which tell the DMA the sources and destinations for the data and how many bytes it is to transfer* (column 23, lines 52-column 24, line 15).

Thus, Baker teaches a subsystem 20 (called processor subsystem) that posts a sequence, command or instruction (called entire command sequence) in the DMA logic (72) "controller" comprising engine (74) for performing (called "setting up") the data transfer (called IDE "integrated drive electronics" data transfer.

Regarding (section viii Arguments) **claim 16** (p. 16 of brief) it is argued that claims 16 requires overwriting only the bits at the bit locations of the register for which a corresponding enable bit in the data value is set with corresponding data bits in the data value. Appellant therefore submits that the arguments submitted herein above in regard to claims 20 appears to be relevant to claim 16.

In response to the above-mentioned argument, particularly, at this outset. Examiner therefore submits that the response to arguments submitted herein above in regard to claims 20 appears to be relevant to claim 16.

Regarding (section viii Arguments) **claim 17-19** (p. 16 of brief) it is argued that claims 17-19 include claim 16 as a base claim. Hence, according to Appellant, claims 17-19 are allowable for at least the reasons stated above in regard to claim 16.

Further, the office action has made applicable the rational of rejection regarding claims 2-3 and 9-10 applicable to the claims 17-19. Appellant therefore submits that the arguments submitted herein above in regard to claims 2-3 and 9-10 appears to be relevant to the claims 17-19.

In response to the above-mentioned argument, particularly, at this outset. Examiner therefore submits that the response to arguments submitted herein above in regard to claims 2-3 and 9-10 appears to be relevant to the claims 17-19.

Regarding (section viii Arguments) **claim 22-23** (p. 17 of brief) it is argued that claims 22-23 include claim 20 as a base claim. Hence, according to Appellant, claims 22-23 are allowable for at least the reasons stated above in regard to claim 20. Moreover, claims 22-23 require issuing a *write command of the data to the location in memory space for the register*. Because, neither Runaldue on column 1, lines 14-41 nor Baker in the abstract teach claim limitation.

In response to the above-mentioned argument, Appellant's recitation of the applied references is noted.

Baker teaches where the PCI-interface ASIC 20 can *issue write commands* on PCI bus 24 by specifying the appropriate *address range* in the controlling *packet control list* (column 20, lines 41-49, where the structure of the *transfer command packet* is shown on Figure 20); PCI slave function 66 *responds to memory write commands* at PCI memory address *ranges specified by base address registers of the* control and status registers (68) (column 20, lines 41-49). The address field that

includes register address bits and individual bit select field addresses bits that will be written with the respective input write data (Fig. 10-11 and respective written description).

Runaldue teaches the enabling selected bits of a data word to be selectively *written into memory* without overwriting *other non-selected bits of the data word stored in the memory* (column 2, lines 3-7); *Selected bits can be written to an address word, without overwriting the unselected bits in the stored word*, by supplying a *mask signal* that *selectively* drives the logic gates of *the selected bits (abstract); t*he logic circuit *selectively causes to overwrite a stored value based upon the write signal and the corresponding bit enable signal and* can be used to access a selected memory location, while *using a mask signal to write selected bits in the address memory word, without overwriting unselected bits of the addressed word.*

Thus, the applied reference(s) teach a *write command of the data to the location in memory space for the register.*

Regarding (section viii Arguments) **claim 24-25** (p. 17 of brief) it is argued that claims 24-25 include claim 16 as a base claim. Hence, according to Appellant, claims 24-25 are allowable for at least the reasons stated above in regard to claim 16. Moreover, claims 24-25 require issuing a write command of the data to the location in memory space for the register. Appellant submits that the arguments submitted herein above in regard to claims 22-23 appear to be relevant to the claims 24-25.

In response to the above-mentioned argument, particularly, at this outset. Examiner therefore submits that the response to arguments submitted herein above in regard to claims 22-23 appear to be relevant to the claims 24-25.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

ZARNI MAUNG
PRIMARY EXAMINER

Conferees:

PATRICE WINDER
PRIMARY EXAMINER

JOHN FOLLANSBEE
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100